Chapter 19 Systems engineering:

19.1 Socio-technical systems

19.2 Conceptual design

19.3 System procurement

19.4 System development

19.5 System operation and evolution

Note: Images excluded due to time constraints

19.1 Socio-technical Systems

Overview

- Socio-technical systems are complex systems that include not just software and hardware, but also people, processes, and organizational policies.
- These systems are deeply embedded in organizational environments and aim to serve a specific business purpose.
- The functioning of each component is intertwined, making the whole system more than the sum of its parts.

Core Characteristics

1. **Emergent Properties**: These are properties that the system as a whole exhibits, rather than individual components. Examples include security and reliability.
2. **Nondeterministic Behavior**: The systems may produce different outputs for the same input due to human factors and new relationships between components.
3. **Subjective Success Criteria**: Success depends not just on the system's functionality but also on organizational objectives and how they are interpreted.

Key Elements Affecting Socio-technical Systems

1. **Process Changes**: A new system may require changes in operational processes, necessitating training and potentially facing user resistance.
2. **Job Changes**: System introduction may alter job roles or deskill workers, causing resistance particularly among professional staff.
3. **Organizational Policies**: The system must align with existing policies, such as privacy rules, necessitating changes either in the system or policies.
4. **Organizational Politics**: The system might affect the power dynamics within the organization.

System Boundaries

- Defining system boundaries is challenging due to multiple influencing factors like organizational culture, laws, and goals.
- Different stakeholders may have varying opinions on where these boundaries should lie.

Challenges and Methodologies

- The complexity and interdependence of sociotechnical factors make it difficult for engineers unfamiliar with social studies to account for them.
- Various sociotechnical systems methodologies exist to assist in understanding the impact of systems on organizations.

Additional Notes

- Properties like reliability, repairability, security, and usability are examples of emergent properties.
- Socio-technical considerations are often crucial in determining whether a system has met its objectives successfully.

19.1.1 Emergent Properties

Overview

- Emergent properties are characteristics that a system as a whole exhibits but cannot be attributed to any specific part of the system.
- These properties often arise from the combination of subsystem properties and their relationships.

Types of Emergent Properties

1. **Functional Emergent Properties**: The purpose of a system that becomes apparent only after its components are integrated. For example, a bicycle becomes a transportation device once assembled.
2. **Nonfunctional Emergent Properties**: These relate to the behavior of the system in its operational environment, such as reliability, performance, safety, and security. Failing to meet a minimum level in these properties often makes the system unusable.

Influencing Factors on Emergent Properties

1. **Hardware Reliability**: Involves the probability of hardware components failing and the time it takes to repair them.
2. **Software Reliability**: Concerned with the likelihood of software components producing incorrect outputs. Unlike hardware, software failures are often transient.
3. **Operator Reliability**: Considers the likelihood of human operators making errors, and how these errors propagate through the system.

Failure Propagation and Context Dependence

- Hardware, software, and operator reliability are interconnected and can influence each other in unpredictable ways.
- A minor failure in one component can escalate into a major system problem, leading to a complete shutdown.
- The reliability of a system is also context-dependent, influenced by its operating environment which cannot always be completely specified by designers.

Measurement and Assessment

- Properties like performance or usability can be assessed once the system is operational.
- Properties like safety and security are not directly measurable and may only be assessed based on undesirable or unauthorized behaviors that occur post-implementation.

19.1.2 Nondeterminism

Overview

- Nondeterminism refers to the unpredictable behavior of a system.
- Software systems running on reliable hardware are often considered deterministic, but sociotechnical systems are inherently nondeterministic due to the involvement of human elements and frequent changes.

Characteristics of Nondeterministic Systems

1. **Human Involvement**: People's responses are influenced by various factors like emotional state, environmental context, and the individual making a request.
2. **System Changes**: Frequent changes in hardware, software, and data contribute to nondeterministic behavior. Interactions between these elements are complex and not fully predictable.

Nondeterminism in Test Inputs

- Using the same set of test inputs at different times may produce different results. This could indicate either system failures or acceptable variations in system behavior, requiring further investigation.

Perception and Adaptability

- Users often perceive the system as nondeterministic because they are not aware of when and why changes have been made.

- Contrary to the notion that nondeterminism is undesirable, it has benefits in sociotechnical systems. It allows the system to adapt its behavior based on its environment, enabling operators to diagnose and recover from problems.

19.1.3 Success Criteria

Overview

- Success criteria for sociotechnical systems are difficult to define due to the complex and evolving nature of "wicked problems" these systems often aim to solve.
- Judgments about the system's success are often based on its effectiveness at the time of deployment rather than the original goals.

Characteristics of Wicked Problems

- **Wicked Problems**: These are complex issues with multiple stakeholders and no definitive problem specification. The true nature of the problem often emerges as solutions are developed.

Complex Goals and Stakeholder Perspectives

1. **Multiple Business Goals**: Systems may be designed to support conflicting business goals, which makes defining success criteria challenging.
    - Example: A healthcare system designed both to improve quality of care and to improve cost-effectiveness led to conflicting goals among doctors and managers.
2. **Shifting Business Environment**: Due to rapidly changing business environments, the goals for which a system was originally designed may no longer be relevant at the time of its deployment.

Regular Reconsideration of Success Criteria

- Success criteria should be **regularly reconsidered** during system development and use as they are not objectively evaluable.
- A system may meet its original requirements but still be ineffective due to changes in the operational environment.

19.2 Conceptual Design

Overview

- **Conceptual Design** is the initial phase in the systems engineering process where the feasibility of an idea is investigated, and an overall vision of the system is developed.
- This phase is essential for communicating the proposed system to nonexperts like senior decision-makers or system users.

Relation to Requirements Engineering

- There is a significant overlap between conceptual design and requirements engineering.

- Activities may include **discussions with potential users**, focus groups, and observations to understand user needs and constraints.

Importance of Vision

- Establishing a vision of the proposed system is crucial and is documented in a **Concept of Operations (ConOps)** document.
- However, the rigid standards of ConOps documents can make them long and unreadable, prompting some to propose a more agile approach.

Conceptual Design Activities

1. **Concept Formulation**: Initial needs are refined to determine what type of system would best meet stakeholder needs.
2. **Problem Understanding**: Discussions with users and stakeholders are essential for comprehending the actual requirements.
3. **System Proposal Development**: Ideas for alternative systems are the basis for a **feasibility study**.
4. **System Structure Development**: An initial architecture can be beneficial for further requirements engineering and assessing feasibility.

System Vision Document

- The output of these activities is the **System Vision Document**, which is critical for senior decision-makers to decide on further development.
- This document is divided into two parts:
    i. A short summary for senior decision-makers outlining the key points and benefits of the proposed system.
    ii. Appendices that elaborate on these ideas in detail for system procurement and requirements engineering.

Use of User Stories

- User stories are effective for conveying the system's capabilities to non-technical stakeholders. They provide a tangible vision of system use and help deliver value to the organization.

19.3 System Procurement

Overview

- **System Procurement** is the decision-making process for acquiring systems from suppliers.
- It involves defining the scope, budget, timescale, and high-level requirements of the system.

Drivers for Procurement

1. **Replacement of Organizational Systems**: To integrate disparate systems or replace costly ones for business benefits.

2. **Regulatory Compliance**: To meet external regulations like Sarbanes-Oxley in the U.S.
3. **External Competition**: For businesses to stay competitive or for military systems to address new threats.
4. **Business Reorganization**: To support new business processes following restructures.
5. **Available Budget**: Determines the scope of the new systems.
6. **Political Factors**: For government systems, political considerations may also drive procurement decisions.

Types of Systems to Procure

1. **Off-the-shelf Applications**: Require minimal configuration.
2. **Configurable Systems**: Require modifications or in-built configuration features.
3. **Custom Systems**: Specially designed and implemented.

Key Issues in Procurement Process

1. **Approved Software Sets**: Organizations often have a recommended set of software, simplifying procurement.
2. **Requirement Mismatch**: Off-the-shelf components may not exactly match the requirements.
3. **Legal Aspects**: For custom systems, the specification is both a legal and technical document.
4. **Public Sector Rules**: Detailed regulations can slow down the procurement process and inhibit agile development.
5. **Contract Negotiation**: Terms, conditions, and possibly requirements changes are negotiated after system selection.

Complexity in Sociotechnical Systems

- Large systems often use a mix of off-the-shelf and custom components.
- **Glueware** may be required for integration, reducing savings from off-the-shelf components.

Consortiums and Principal Contractors

- For large systems, a consortium of suppliers might bid for the contract.
- A **principal contractor** usually coordinates the project, integrating subsystems built by subcontractors.

Impact of Poor Procurement Decisions

- Wrong choices can lead to late delivery, unsuitable systems, and increased complexity in system and software engineering.
- Decisions can profoundly affect the **security and dependability** of a system.

Political vs Technical Decisions

- Procurement decisions are often influenced by **political factors** rather than technical merits, complicating the engineering process.

19.4 System Development

Overview

- **System Development** is the multifaceted process of creating a system through development or procurement, and integrating these elements.
- The process involves multiple overlapping activities, starting from high-level functional and nonfunctional system requirements.

Development Model

- Systems engineering generally follows a **waterfall process model**.
- The model is plan-driven because system elements are developed independently and may involve different contractors.

Fundamental Development Activities

1. **Requirements Engineering**: Refines, analyzes, and documents high-level and business requirements.
    - Covered extensively in Chapter 4.
2. **Architectural Design**: Establishes the system's overall architecture and its components.
    - Overlaps with requirements engineering.
3. **Requirements Partitioning**: Assigns responsibilities for system requirements to various subsystems.
    - Focuses on hardware, software, and operational processes.
4. **Subsystem Engineering**: Involves software development, hardware configuration, and defining operational processes.
5. **System Integration**: Combines system elements to produce the final system.
    - Emergent system properties become apparent here.
6. **System Testing**: Comprehensive testing of the whole system.
    - May involve both developer testing and user acceptance testing.
7. **System Deployment**: Making the system available to users.
    - Involves data transfer and system communication setup.

Spiral Model for Requirements and Design

- Requirements and design are developed concurrently in a **spiral model**.
- The model reflects the reality that requirements and design are interlinked and affect each other.

Integration Approaches

- **Incremental Integration** is often the best approach for combining subsystems.
- This method is efficient for error location and scheduling.

Challenges and Complexities

- **Costs**: Modifying large off-the-shelf systems like ERP systems can be expensive.
- **Testing Budget**: Running out of time or budget during testing can lead to error-prone systems.
- **Contractor Disputes**: Can arise when issues occur in subsystem interactions.
- **Deployment Challenges**: Adapting the system to unexpected user environments can be problematic.

Organizational and Political Influences

- Solution choices may be influenced by wider organizational and political considerations, such as preference for national suppliers.

Final Stage: System Delivery and Deployment

- Involves system configuration, data transfer, and preparation of user documentation and training.
- Often takes longer and costs more than anticipated due to unforeseen challenges.

19.5 System Operation and Evolution

Overview

- **Operational processes** are the protocols involved in utilizing the system as designed.
- Challenges may arise at this stage if the system's functions do not align with real operational needs.

User Adaptation and Domestication

- Users go through a period of **"domestication"** where they adapt to the new system and figure out practical ways to use it.
- **User Interface Design**: Should not only cater to inexperienced users but also be adaptable for experienced users who may prefer quick rather than easy ways of using the system.

Automation vs Human Involvement

1. **Increased Technical Complexity**: Automating systems to minimize human involvement can increase costs and time, as the system must be designed to cope with all possible failure modes.
    - Provision for human intervention still needed for unanticipated failures.
2. **Human Adaptability**: Humans can handle unexpected situations, allowing for more flexible system design.
    - Operators can often recover from unexpected events by using the system in nonstandard ways.

Designing Operational Processes

- **Operational processes** should be designed to be flexible and adaptable.
- They should allow for operations to be done in any order and not be overly constraining.
- Operators often improve the process based on their real-world experiences.

System Compatibility Issues

- Problems can arise when the new system operates alongside existing systems.
- Issues may include physical incompatibility, data transfer difficulties, and increased operator error due to different user interfaces.

Conclusion

- It's crucial to design systems and operational processes that are adaptable and consider both the limitations and advantages of human involvement.

- Issues like system compatibility and real-world operational needs often become apparent only after the system is in operation.

19.5.1 System Evolution

Overview

- Large, complex systems often have long lifetimes and are subject to **evolution** to meet new requirements or adapt to changes.
- **System evolution** involves revisiting the development process to update hardware, software, and operational processes.

Reasons for System Longevity

- **Investment Cost**: High initial costs necessitate long-term value delivery.
- **Loss of Expertise**: Organizations may lose the skill to specify new system requirements.
- **Replacement Cost**: Extremely high, justifiable only if significant cost savings are achievable.
- **Return on Investment**: New systems may not offer higher ROI than maintaining existing ones.
- **Risks of Change**: Replacing systems carries inherent operational risks.
- **System Dependencies**: Systems are interconnected; changing one could necessitate changes in others.

Challenges in System Evolution

1. **Business and Technical Analysis**: Changes must align with business goals and not just be technically motivated.
2. **Subsystem Interdependencies**: Changes in one subsystem may adversely affect others, requiring further adjustments.
3. **Unrecorded Design Decisions**: Lack of documentation may make it difficult to understand why specific choices were made.
4. **Increasing Costs Over Time**: As systems age, their structure gets corrupted, making further changes more costly.

Legacy Systems

- Often rely on obsolete technology and include **legacy processes** that are difficult to change.
- Changing one part usually involves changes to other interconnected components.

Risks and Vulnerabilities in Evolution

- Changes made during evolution can introduce **security and dependability risks**, especially if those implementing changes are not the original developers.
- Complete retesting may be impossible due to costs, leaving potential for unnoticed adverse effects.

Conclusion

- System evolution is a complex, costly, but necessary aspect of maintaining large, long-term systems.

- Challenges include business alignment, subsystem dependencies, and risk management, especially for legacy systems.

Summary

- Systems engineering is concerned with all aspects of specifying, buying, designing, and testing complex sociotechnical systems.
- Socio-technical systems include computer hardware, software, and people, and are situated within an organization. They are designed to support organizational or business goals and objectives.
- The emergent properties of a system are characteristics of the system as a whole rather than of its component parts. They include properties such as performance, reliability, usability, safety, and security.
- The fundamental systems engineering processes are conceptual systems design, system procurement, system development, and system operation.
- Conceptual systems design is a key activity where high-level system requirements and a vision of the operational system is developed.
- System procurement covers all of the activities involved in deciding what system to buy and who should supply that system. Different procurement processes are used for off-the-shelf application systems, configurable COTS systems, and custom systems.
- System development processes include requirements specification, design, construction, integration, and testing.
- When a system is put into use, the operational processes and the system itself inevitably change to reflect changes to the business requirements and the system’s environment.